III.A.2.3. Core ONCOCIN Research

ONCOCIN is a data management and therapy advising program for complex cancer chemotherapy experiments. The development of the system began in 1979, following the successful generalization of MYCIN into the EMYCIN expert system shell. The ONCOCIN project has evolved over the last eight years: the original version of ONCOCIN ran on the time-shared DECSystem-20 computers using a standard terminal for the time-oriented display of patient data. The current version uses compact workstations running on the Ethernet network with a large bit-mapped displays for presentation of patient data. The project has also expanded in scope. There are three major research components: 1) ONCOCIN, the therapy planning program and its graphical interface; 2) OPAL, a graphical knowledge entry system for ONCOCIN; and 3) ONYX, a strategic planning program designed to give advice in complex therapy situations. Each of these research components has been split into two parts: continued development of the cancer therapy versions of the system, and generalization of each of the components for use in other areas of medicine. This portion of the annual report will concentrate on the *generalization tasks* related to treatment planning, knowledge acquisition, and research to extend ONCOCIN for application in clinical trial domains other than medical oncology (E-ONCOCIN).

In addition, we will discuss the continued development of a generalized knowledge acquisition tool designed to encode descriptions of clinical trials. The system, named PROTÉGÉ, was the Ph.D. thesis work of Mark Musen, (who joined our faculty this year). The output of PROTÉGÉ is an OPAL-like input system designed for a target clinical area such as hypertension. This input system (HTN-OPAL) can then be used to create the hypertension knowledge base for an E-ONCOCIN like system. This experiment was carried out this year for both the hypertension and oncology domains. Details of this project are described later in this report.

(1) Overview of the ONCOCIN Therapy Planning System

ONCOCIN is an advanced expert system for clinical oncology. It is designed for use after a diagnosis has been reached, focusing instead on assisting with the management of cancer patients who are receiving chemotherapy. Because anticancer agents tend to be highly toxic, and because their tumor-killing effects are routinely accompanied by damage to normal cells, the rules for monitoring and adjusting treatment in response to a given patient's course over time tend to be complex and difficult to memorize. ONCOCIN integrates a temporal record of a patient's ongoing treatment with an underlying knowledge base of treatment protocols and rules for adjusting dosage, delaying treatment, aborting cycles, ordering special tests, and similar management details. The program uses such knowledge to help physicians with decisions regarding the management of specific patients.

A major lesson of past work in clinical computing has been the need to develop methods for integrating a system smoothly into the patient-care

environment for which it is intended. In the case of ONCOCIN, the goal has been to provide expert consultative advice as a by-product of the patient data management process, thereby avoiding the need for physicians to go out of their way to obtain advice. It is intended that oncologists use ONCOCIN routinely for recording and reviewing patient data on the computer's screen, regardless of whether they feel they need decision-making assistance. This process replaces the conventional recording of data on a paper flowsheet and thus seeks to avoid being perceived as an additive task. In accordance with its knowledge of the patient's chemotherapy protocol, ONCOCIN then provides assistance by suggesting appropriate therapy at the time that the day's treatment is to be recorded on the flowsheet. Physicians maintain control of the decision, however, and can override the computer's recommendation if they wish. ONCOCIN also indicates the appropriate interval until the patient's next treatment and reminds the physician of radiologic and laboratory studies required by the treatment protocol.

(2) Implementation of the ONCOCIN Workstation in the Stanford Clinic

In mid-1986, we placed the workstation version of ONCOCIN into the Oncology Day Care clinic. This version is a completely different program from the version of ONCOCIN that was available in the clinic from 1981-1985 — using protocols entered through the OPAL program, with a new graphical data entry interface, and revised knowledge representation and reasoning component. One person in the clinic (Andy Zelenetz) became primarily responsible for making sure that our design goals for this version of ONCOCIN were met. His suggestions included the addition of key protocols and the ability to have the program be useful for clinicians as a data management tool if the complete treatment protocol had not yet been entered into the system. Additional fellows were trained on a very stable release of ONCOCIN that became available in early 1988. A version of the system was sent to the University of Pittsburgh for evaluation and to the National Library of Medicine Artificial Intelligence Demonstration Center. For these various efforts, Janice Rohn has created an extensive user manual, sample patient interactions, and reminder cards to shorten the training period for ONCOCIN.

The process of entering a large number of treatment protocols in a short period of time led to other research topics including: design of an automated system for producing meaningful test cases for each knowledge, modification of the design of the time-oriented data base and the methods for accessing the data base, and the development of methods for graphically viewing multiple protocols that are combined into one large knowledge base. These research efforts will continue into the next year. In addition, some of the treatment regimens developed for the original mainframe version are still in use and can be transferred to the new version of ONCOCIN.

We also received new insights about the design of the internal structures of the knowledge base (e.g., the relationship between the way we refer to

chemotherapies, drugs, and treatment visits). We will continue to optimize the question-asking procedure, the method for traversing the plan structure in the knowledge base, and consider alternative arrangements used to represent the structure of chemotherapy plans. Although we have concentrated our review of the ONCOCIN design primarily on the data provided by additional protocols, we know that non-cancer therapy planning problems raise similar issues. The E-ONCOCIN effort is designed to produce a domain-independent therapy planning system that includes the lessons learned from our oncology research.

(3) E-ONCOCIN: Domain Independent Therapy Planning

During the past two years, our E-ONCOCIN research has concentrated on understanding how protocols in medicine vary across subspecialties. We are examining several application areas: the intensive care unit, insulin treatment for diabetes, hypertension protocols, and both standard and complex cancer treatment problems. The diagnosis and therapy selection for patients in the intensive care unit is a natural application area because it is based on changing data and the need to determine the response to therapy interventions. In addition, it is an area where reasonable mathematical models of the respiratory system can be integrated into the expert system (see description of the VentPlan system). We also felt that the area of insulin treatment for diabetes would be a good area to explore. Like cancer chemotherapy, the treatments for diabetes continues over a long period of time and has been the area of intensive protocol development. Unlike cancer chemotherapy, the treatment plan must handle multiple treatments in one day and deemphasises the use of multiple drugs (although there are a variety of types of insulin). During 1987, using the medical literature and several internists in the medical computer science research group (Mark Frisse, Mark Musen, and Michael Kahn), we performed knowledge acquisition experiments for insulin treatment of diabetes. The proposed structure for the knowledge base was implemented using the object-oriented programming language upon which ONCOCIN has been based. These experiments, like those of adding more protocols to ONCOCIN, demonstrated the need for changes in the way that the knowledge base can access the time-oriented data base that records patient data and previous conclusions. The relationships between the different doses and types of insulin treatments will also require alternative ways of building treatment hierarchies. Thus, our initial experiments have shown that many of the elements of the ONCOCIN design are sufficiently general for other application areas, but that some specific elements (particularly the representation of temporal events) will have to be generalized. A description of our revised temporal representations has appeared in a thesis by Michael Kahn, who is at Washington University in St. Louis, based on work completed at Stanford and U.C.S.F.

A logical extension of Kahn's work has been an investigation of how to modify the EONCOCIN framework so that it can work with established data base tools instead of the hand-tailored data base currently in use. In making this

transformation, we need to maintain the access to data that is mediated by the temporal network, although most relational data bases are not organized for encoding temporal information. In this work, we must be able to describe the changing clinical context and event intervals that show up in many diverse application areas. An example of a new area that we are exploring is the treatment of AIDS patients on clinical protocols. While this area is similar to some aspects of oncology protocols, we are faced with significant differences in the way that treatment is delivered. AIDS patients do not always follow the type of strict temporal schedules (e.g., regular visits to outpatient clinics) seen with oncology patients. They have a chronic disease with acute exacerbations of opportunistic infections. Medications are often given orally as opposed to the controlled intravenous infusions of medical oncology patients. Furthermore, the medication schedule is interrupted by frequent hospitalizations and confounded by taking drugs not on the protocol. Together, these factors will require a much more flexible model of the temporal dimension of treatment planning.

(4) OPAL: Graphical Knowledge Acquisition Interface

OPAL is a graphical environment for use by an oncologist who wishes to enter a new chemotherapy protocol for use by ONCOCIN or to edit an existing protocol. Although the system is designed for use by oncologists who have been trained in its use, it does not require an understanding of the internal representations or reasoning strategies used by ONCOCIN. The system may be used in two interactive modes, depending on the type of knowledge to be entered. The first permits the entry of a graphical description of the overall flow of the therapy process. The oncologist manipulates boxes on the screen that stand for various steps in the protocol. The resulting diagram is then translated by OPAL into computer code for use by ONCOCIN. Thus, by drawing a flow chart that describes the protocol schematically, the physician is effectively programming the computer to carry out the procedure appropriately when ONCOCIN is later used to guide the management of a patient enrolled in that protocol.

OPAL's second interactive mode permits the oncologist to describe the details of the individual events specified in the graphical description. For example, the rules for administering a given chemotherapy will vary greatly depending upon the patient's response to earlier doses, intercurrent illnesses and toxicities, hematologic status, etc. For example, one form permits the entry of an attenuation schedule for an agent based upon the patient's white count and platelet count at the time of treatment. Tables such as this are generally found in the written version of chemotherapy protocols. Thus OPAL permits oncologists to enter information using familiar forms displayed on the computer's screen. The contents of such forms are subsequently translated into rules and other knowledge structures for use by ONCOCIN.

Status of the OPAL System

The OPAL is one of the few graphical knowledge acquisition systems ever designed for expert systems. Even fewer are designed to be used as the main method for entering knowledge as opposed to a proof of concept implementation. We have pursued three directions in the development of the OPAL system, also in response to the large number of protocols entered through this system during the several years.. The first direction is the modification of graphical forms needed to allow the entry of facts that did not show up in the protocols used to test the initial version of OPAL. OPAL continues to assume that most of the knowledge to be entered will have very stereotyped forms, e.g., dose attenuations for most treatment toxicities are based on a comparison of only one laboratory measurement at a time, such as using the BUN to adjust for renal toxicity. We sometimes need much more complex ways of stating the scenarios in which dose adjustments may be necessary. This need has led us in a second direction, towards a "lower-level" rule entry approaching the syntax of the reasoning component of ONCOCIN, but using graphical input devices where applicable. A major accomplishment of this last year was to experimentally combine the OPAL and ONCOCIN programs into one working program, and to completely enter knowledge from OPAL using both the high level tools and lower level rule editors, but without needing to make changes at the ONCOCIN side of the system. The OPAL program maps the information provided on the graphical forms into a complex data structure (called the IDS) that represents the required knowledge to specify the contents of a protocol. This data structure is used for copying information from one protocol to another, and as the basis for the creation of the ONCOCIN knowledge base. Our experiments with OPAL, and our intention to generalize OPAL use outside of oncology protocols, suggests that we reorganize the OPAL program to use a relational data base to store its knowledge. We have patterned the data base after an existing data base query syntax. Because no data bases exist for the InterLisp language upon which OPAL is based, we reimplemented the data base from its written description. We continue to explore the appropriate avenue for the connection of our knowledge acquisition systems to data bases, and have concentrated on the SQL query language to a relational data base using the client-server model (e.g., the physical data base may exist on a different machine than the knowledge acquisition tool — transmitting the query and the response over the network).

With the future of dedicated lisp processors looking very unclear, we began to explore alternative platforms for developing the interface for OPAL-like systems. We have begun experiments using HyperCard on the Mac II and Interface Builder on the NeXT machine. In order to build experience with the each of these possible platforms, we have re-implemented portions of OPAL system, and are analyzing the results. It is particularly hard to determine the best platform since the NeXT machine software is still in a rudimentary

stage, and HyperCard on the Mac II has significant limitations including small "card size" and the inability to display multiple cards simultaneously.

(5) Generalized Knowledge Acquisition through PROTÉGÉ

Mark Musen designed and implemented the first version of the PROTÉGÉ knowledge-acquisition-system development tool. PROTÉGÉ is used to collect information which describes the concepts (both entities and their relationships) in an application area for which a skeletal-planning type of expert system would be useful, concentrating on clinical trials. The system acquires the "ontology" of a domain through a series of fill-in-the-blank forms and a "flowchart-entry" tool. These concepts are then mapped onto a set of generic forms, which, in turn, create a knowledge acquisition tool for the application area.

PROTÉGÉ makes use of the forms management system built for the original OPAL, and a newly developed relational data base management system written for the Xerox InterLisp-D workstations. The output of PROTÉGÉ is an OPAL-like set of forms tailored to the special structures of the application area. To test these ideas, we first reimplemented portions of the OPAL interface from a high level description of oncology. After the translation process to the ONCOCIN reasoning program, a consultation was run that matched the manually built system. This experiment was then repeated for the area of hypertension protocols for which ONCOCIN had never been specifically designed. With some minor generalizations to the ONCOCIN reasoner and interviewer, we were able to run a hypertension consultation. With Mark returning as a faculty member, this work has continued this year, but faces the same platform and basic tool issues as described above. Portions of the PROTÉGÉ interface is being reimplemented using the Interface Builder on the NeXT computer.

(6) Speech Input to Expert Systems

(6.1) Prototype Speech Hardware/Software System

In 1987 we began a project to explore the integration of speech-recognition technology into the interface to ONCOCIN running on the XEROX Lisp workstations. The project uses a commercially available continuous-speech-recognition product loaned by the vendor, Speech Systems, Inc. (SSI) of Tarzana, California. The speech recognizer consists of a custom processor, called the Phonetic Engine® and a suite of software modules called the Phonetic Decoder™. The Phonetic Decoder™ initially ran on a SUN 3/75 and now runs on the NeXT computer.

The development of this project requires significant experience in distributed computing since the phonetic device, initial parsing software, and the ONCOCIN system all reside on different pieces of hardware. One of the early steps is to allow the Lisp machine to remotely control the parsing software on the SUN. We built an interpreter for communicating between the speech software library running on the SUN and the Xerox Lisp machine. This

interpreter reads Lisp-style function calls corresponding to the speech library routines and returns Lisp-style results as remote procedure call (RPC) mechanism. We then wrote the library of corresponding Lisp stub routines and a function to connect to the SUN workstation and start the server. In normal operation, we call the C-based speech library routines from Lisp as if they were Lisp functions. During this year, we were able to port a version of the SSI system to the NeXT machine. In addition, we mounted a new version of the speech hardware (PE200) such that we can compare the two versions of the hardware with each other in terms of accuracy, speed, and ease of use. In addition, we were able to obtain a copy of the CMU speech understanding system (running on the NeXT machine), and were able to make some comparisons with the SSI speech processing hardware. All of these systems require extremely fast processors in order to deliver reasonable response time. The announcement of relatively inexpensive and fast RISC-based architectures should enhance the acceptance of speech input systems. Because some of the applications of the speech equipment do not require continuous speech, we are also evaluating this mode.

We created a prototype system that permits users to navigate the graphical interface and enter clinical data using speech. The system uses the location of the cursor on the screen to provide a context for choosing candidate grammars with which to attempt recognition of a user's utterance. The system dynamically re-orders the list of candidate recognition grammars based on the dialog history. Albeit with limitations on the legal grammars, it is now possible to carry on most of the ONCOCIN data acquisition steps using speech alone or speech plus pointing with the mouse. In addition, some elements such as the neural toxicities can be entered as textual descriptions and automatically encoded as one the 1-4 point scale used on flowsheet forms. This system was extended such that it was possible to have an entire ONCOCIN consultation only with voice commands.

In order to translate an utterance back into an action that can occur in the ONCOCIN interface, we need the ability to reparse the text string returned by the SSI equipment. The SSI equipment uses (potentially complex) syntax, built up of various classifications, to understand sentences but returns just the ASCII component of the actual sentence; you can not get it in terms of the original classifications in the syntax (which are generally semantically significant). When the ASCII string is returned, a description of the syntax is used to convert the string into a parse tree that relates directly to the grammar definition. We can now process the returned information at a much higher level than was possible with the simple ASCII text. In addition, we have built tools to perform the semantic analysis that is converted to specific actions in the interface language. For example, the utterance "white blood count is 3.2" is parsed using a grammar "<parameter> is <value>." When the string is recognized, it must be turned back into the action sequence — in this case, opening the hematology form if it is not already open, and highlighting the WBC row, and placing the value 3.2 in the column corresponding to the current visit. While the graphical effects appear simple,

these internal transformations may be quite complex. We are deriving from this experience a description of an interface manipulation language that details the "legal operations" in an ONCOCIN-like graphical interface.

We have also explored a second medical record-keeping task — the creation of portions of a progress note that describes in textual form the changes in the patients status from week to week. We have developed a system that uses broad categories of data as specified on the spreadsheet as a prompt for textual input. For example, the spreadsheet includes a line for "gastrointestinal toxicity" of grades 1 to 4. This becomes the topic of a sentence to be included in the progress report such as "The patient has experienced nausea and vomiting one to three times per day over the last week." The number of sentences that are possible cannot practically be displayed on the screen, so we are experimenting with various types of prompting to give the user a sense of what sentences the system will be able to recognize; e.g., by selecting random examples from the grammars to present as hints for entry or by presenting a graphical version of the legal syntax at any point. This system was further extended to develop a portion of the physical examination (PE) portion of the progress note. The PE consists of a short description about each organ system in the body, concentrating on those specific aspects that are remarkable during a patient visit (changes or significant findings). Using hypertext techniques similar to those found in outlining programs such as MORE, we created a hierarchical description of the breast portion of the PE, and have been able to create reasonable interactions for this one segment of the PE. As the user becomes more familiar with the structure of the legal syntax and the descriptions necessary to reach a particular level of detail, then they can use speech to bypass the graphical interface and directly enter the utterance. We are exploring ways to scale these techniques up to allow for entry of the entire PE, while examining how to organize the less structured elements of the progress note.

(6.2) Speech Experiments

We are performing experiments to (1) enhance the system's grammars with a wider range of phrases clinicians actually use when talking to a computer and (2) gain insights into clinicians' models of spoken interaction with advice systems so that we may ground our interface design in observed practice. In order to assess how physicians would speak to a computer in an ideal situation without constraints or prior assumptions, we are conducting a series of experiments which simulate continuous-speech understanding by computers. The setting of these experiments includes a hidden computer operator simulating the output of ONCOCIN if it had the ability to understand the spoken input, as well as a video camera to record both audio and visual clues. Typed responses from the operator are translated back as actions on the computer display as well as audio responses through the use of a speech synthesizer. It appears to the subject as if the computer is understanding and responding to their speech. The physicians use

ONCOCIN in the same manner as it is used in the clinic when they see patients, but with the added capability of speech input.

These experiments enable us to both build up a basic vocabulary for the speech system as well as examine subtle linguistic issues to guide future directions. The experiments have been completed and we are analyzing the data in order to describe user-specific grammars, and to see how individuals react to purposeful misunderstandings by the computer. We experimenting we a range of possible misunderstandings (e.g., from "What did you say?" to "The platelet count was WHAT?"). Our initial impressions of these experiments were that each subject quickly developed her own subgrammar for entering the information, often based on their belief about the system's knowledge of the domain, and that subjects responded directly to misunderstandings with partial phrases in a manner similar to being asked by a human. This experiment underscores the importance of being able to obtain not only the top-rated sentence from the recognition system, but also an indication as to which parts of the utterance were most likely to have been misrecognized.

(7) Object Language Support for ONCOCIN Project

We released a new version of our object language at the start of this past year which has proven to be the most stable and powerful version to date. There have been a number of minor bug fixes and several feature additions over the course of the year but for the most part the system as required much less attention than in previous years. The number of new systems being built on it (like our speech work) continues to increase. Future planning for the system consists of determining whether or not it should be converted to Common Lisp, based on whether object systems available under Common Lisp are sufficient for our needs, and if we do convert it what it would it look like if properly integrated with that language.

(8) Personnel

Samson Tu has been primarily responsible for the design of E-ONCOCIN, Michael Kahn developed the temporal representations used by the system, Clifford Wulfman has been involved with extensions the the data entry interface and the extensions to the interface in order to add speech input. Samson and Cliff were responsible for extensions to their programs to support the PROTÉGÉ effort. David Combs has been involved with the knowledge acquisition interface and provided major programming support for the PROTÉGÉ effort. Janice Rohn has been involved with the entry of protocols, interaction with physicians using the system, documentation of the system, and execution of the speech experiments. Christopher Lane has developed the object-oriented systems software upon which the entire ONCOCIN system is designed. He has be instrumental in developing the systems software for the speech project, and has performed our evaluations of different speech input devices and configurations. . Ellen Isaacs, a Ph.D. student in Psycholinguistics has helped to design the simulated speech-input

experiments. Monica Rua has developed the progress note software in conjunction with Cliff and Christopher.

III.A.2.2. Core AI Research

(1) Rationale

Artificial Intelligence (AI) methods are particularly appropriate for aiding in the management and application of knowledge because they apply to information represented symbolically, as well as numerically, and to reasoning with judgmental rules as well as logical ones. They have been focused on medical and biological problems for well over a decade with considerable success. This is because, of all the computing methods known, AI methods are the only ones that deal explicitly with symbolic information and problem solving and with knowledge that is heuristic (experiential) as well as factual.

Expert systems are one important class of applications of AI to complex problems — in medicine, science, engineering, and elsewhere. An expert system is one whose performance level rivals that of an human expert because it has extensive domain knowledge (usually derived from an human expert); it can reason about its knowledge to solve difficult problems in the domain; it can explain its line of reasoning much as an human expert can; and it is flexible enough to incorporate new knowledge without reprogramming. Expert Systems draw on the current stock of ideas in AI, for example, about representing and using knowledge. They are adequate for capturing problem-solving expertise for many bounded problem areas. Numerous high-performance, expert systems have resulted from this work in such diverse fields as analytical chemistry, medical diagnosis, cancer chemotherapy management, VLSI design, machine fault diagnosis, and molecular biology. Some of these programs rival human experts in solving problems in particular domains and some are being adapted for commercial use. Other projects have developed generalized software tools for representing and utilizing knowledge (e.g., EMYCIN, UNITS, AGE, MRS, BB1, and GLisp) as well as comprehensive publications such as the three-volume Handbook of Artificial Intelligence[1] and books summarizing lessons learned in the DENDRAL and MYCIN research projects[2].

This report documents progress on the basic or core research activities within the Knowledge Systems Laboratory (KSL), funded in part under the SUMEX resource as well as by other federal and industrial sources. This work explores a broad range of basic research ideas in many application settings,

---

1 Barr, A., Cohen, P. R., and Feigenbaum, E. A. *The Handbook of Artificial Intelligence, Volumes I, II, and III.* William Kaufmann, Inc., Los Altos, CA, 1981 and 1982.

2 Buchanan, B. G., and Shortliffe, E. H. Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley, Reading, MA, 1984.

Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A., and Lederberg, J. Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project. McGraw-Hill, New York, NY, 1980.

all of which contribute in the long term to improved knowledge based systems in biomedicine.

(2) Highlights of Progress

In the last year, research has progressed on several fundamental issues of AI. As in the past, our research methodology is experimental; we believe it is most fruitful at this stage of AI research to raise questions, examine issues, and test hypotheses in the context of specific problems, such as management of patients with Hodgkin's disease. Thus, within the KSL we build systems that implement our ideas for answering (or shedding some light on) fundamental questions; we experiment with those systems to determine the strengths and limits of the ideas; we redesign and test more; we attempt to generalize the ideas from the domain of implementation to other domains; and we publish details of the experiments. Many of these specific problem domains are medical or biological. In this way we believe the KSL has made substantial contributions to core research problems of interest not just to the AIM community but to AI in general.

Progress is reported below under each of the major topics of our work. Citations are to KSL technical reports listed in the publications section.

(2.1) Large Multi-use Knowledge Bases for Science and Engineering

There is considerable power in the current stock of AI techniques, as exemplified by the rate of transfer of ideas from the research laboratory to commercial practice. But we also believe that today's technology needs to be augmented to deal with the complexity of medical information processing. One of our core research goals is to analyze the limitations of current techniques and to investigate the nature of methods for overcoming them. Long-term success of computer-based aids in medicine and biology depend on improving the programming methods available for representing and using domain knowledge. That knowledge is inherently complex: it contains mixtures of symbolic and numeric facts and relations, many of them uncertain; it contains knowledge at different levels of abstraction and in seemingly inconsistent frameworks; and it links examples and exception clauses with rules of thumb as well as with theoretical principles. Moreover, strategies for using domain knowledge can be complex as well, particularly in dynamic environments which require continual reassessment of the best use of limited computational resources. Current techniques have been successful only insofar as they severely limit this complexity. As the applications become more far-reaching, computer programs will have to deal more effectively with richer expressions, more voluminous amounts of knowledge and more complex control strategies.

Expert systems are being developed that impact nearly every field of human endeavor: medicine, manufacturing, financial services, diagnosis of machinery, geology, molecular biology and structural design, to name a few. Each new instance is a confirmation of the Knowledge Principle (knowledge is power). In each system, expert level problem-solving performance is obtained

by using relatively simple and uniform reasoning methods which access an extensive body of domain knowledge. The ability of these systems lies primarily in the *specific* concepts, facts, methods, models, etc. that can be brought to bear on the problem. A corollary to the Knowledge Principle is that significant improvements in the power of knowledge-based systems will be derived primarily from the ability to access large amounts of knowledge.

To test that corollary, we are embarked on a multi-year research effort that will develop methods for building *large, multi-use knowledge bases* (LMKB). We believe construction of a LMKB is an essential step toward resolving two fundamental problems plaguing the current generation of expert systems. The first is *brittleness*: current systems can exhibit only a very narrow range of expert behavior, and their performance falls off precipitously at the limits of their expertise. The second problem is *over-specialization*: a knowledge base constructed to support of one type of expert task (e.g., diagnosis) cannot be used to support other types of tasks (e.g., design).

Our hypothesis is that both the problems of brittleness and over-specialization can be addressed by constructing large, multi-use knowledge bases. A LMKB would:

1) encode domain knowledge in greater depth and breadth than required for any specific task,
2) encode knowledge that cuts across many domains of expertise, and
3) serve as a core repository of knowledge to be accessed by large numbers of specific applications.

Research of this scope raises many important research issues. Of primary importance are issues of *knowledge representation*. The AI field needs to broaden its understanding of how to give programs a representation of the physical world, its phenomena, its processes, and its devices (in addition, of course, to the conventional mathematical/numeric representations of scientific computing). These issues are both epistemological and technological (how the concepts are named, described, and related; and how they are stored for efficient access and use by reasoning processes). Of these, the epistemological issues are key, because we intend our knowledge bases to be long-lasting, robust, and built upon by many other groups. The "large" in large knowledge bases cannot become a reality unless there is cumulation, and cumulation will only come about if we demonstrate the epistemological adequacy of our pioneering efforts at representing the world of physics and engineering. During the past year we have been exploring a variety of representations and the systems which employ them, including CYC from MCC, CLASS from Schlumberger, and QPE from Univ. of Illinois.

A second major issue is one we call *knowledge compilation*. This is the bridge between the second-era systems and the expert systems of the first era. Effective problem solving is not typically carried out at the level of first principles, but rather at the level of more compact, efficient forms of knowledge, compiled from experience with specific tasks. So-called

knowledge compilers are needed to translate from the more general forms of knowledge (e.g. the basic principles of the physical science and engineering) to highly specific forms needed for diagnosis, design, etc. During the initial year of this project, we demonstrated the feasibility of this approach by deriving a set of diagnostic rules and a set of redesign heuristics from a single knowledge base containing a model of the structure and behavior of the Reaction Wheel Assembly mentioned above. We are also developing an integrated scheme for using "first principles" knowledge of the physical world for simulation. Given a description of the structure of a device in terms of its constituent objects and their relations, the system identifies applicable physical laws, processes, types of matter, etc. and produces a set of equations to describe the behavior of the device. The equation model is analyzed using the method of causal ordering to produce a model that reveals the dependency relations among the parameters of the model[1].

Closely related to knowledge compilation is its inverse, *knowledge justification*. That is, how can one justify the highly specialized knowledge in a typical task-specific expert system in terms of more fundamental laws of nature and/or principles of engineering? Our approach here is to treat both the compilation and justification processes together. As knowledge is compiled into a more specialized form, the system will record the links back to the more fundamental sources of knowledge. A side benefit of knowledge justification will be the ability to provide more satisfactory explanations of an expert system's line of reasoning than just a trace of the rules that were fired.

A fourth research issue is concerned with *model-based reasoning*, and in particular, reasoning about physical devices using multiple approximate models. Scientists and engineers have the ability to model the physical world at different levels of detail. For example, we can solve problems of motion in a gravitational field by assuming objects of no size (point masses) moving in a vacuum. This is an appropriate model for predicting the fall-time of a rock but not of a feather. Having a detailed model of the domain under consideration allows one to reason correctly in a wide range of situations, including previously unanticipated ones. However, this robustness may entail an excessive computational cost. A diagnostic program that used a detailed model of the human metabolic system, for example, would be a very powerful tool but may be prohibitively costly to run. How does one select the appropriate model? What solution method should be used with each model? How can a reasoning system recognize that it must use an alternative model, and how does one shift from one model to another? These issues are currently being investigated in the context of the electrical power system on the Hubble Space Telescope (HST). During the past year we have begun to investigate methods of controlling the expense of model-based reasoning by generating multiple, approximate models of the task domain, and developing methods for choosing, from this space of possible models, the simplest model

---

[1] Iwasaki, Y., and Simon, H. A. "Causality in Device Behavior." *AI Journal,* 29:3-32, 1986.

that provides acceptable answers. Our proposed approach to tackling this problem is to explicitly represent the assumptions underlying each model, and to use these explicitly represented assumptions to help pick the appropriate model.

Finally, we must learn to integrate different types of reasoning in physical domains. For example, *case-based reasoning* is widely used in several engineering domains, especially in the design of new artifacts. Designers search for a previous design that most closely matches the current requirements, and then make appropriate modifications. However, even when past cases are used to construct a new solution, model-based reasoning is needed to make the modifications to fit the current problem, and also to detect unforeseen effects of those modifications..

(2.2) Adaptive Intelligent Systems

How can we design flexible control structures for powerful problem solving programs? How can we use these structures effectively in many problem domains? How can we represent processes and reason about their behavior, and perform intelligent actions under *real-time* requirements?

We have continued to develop the BB1 blackboard architecture to address these and related problems. In particular, we have begun or continued work on the following domain-independent BB1 modules:

- The Focus module provides a dynamic focus of attention and protects the application system from input data overload. It continuously monitors all sensors, abstracts sensed data (e.g., as value categories, averages, trends, or patterns), and filters the abstracted data before relaying it to the BB1 application system.
- The ReAct module provides time-sensitive problem detection and response capabilities. It propagates asynchronously sensed data through a hierarchically partitioned belief network. It uses time and other resource constraints to determine whether to continue the analysis or act on the basis of its current assessment.
- The ICE module provides reasoning from first principles to handle complex or unfamiliar problems. It uses structure/function representations of generic physical systems (e.g., flow, diffusion) and particular domain systems (e.g., respiration, circulation) to diagnose, explain, and predict problems.
- The TPlan module provides time-sensitive planning of coherent courses of action. It plans actions forward in time and at successively more detailed levels of abstraction. It uses time and other resource constraints to determine whether to continue plan refinement or act on the basis of its current plan.
- The TDB module provides a temporally organized database of observed, expected, and intended models of external entities, and associated temporal reasoning functions.

In collaboration with Dr. Adam Seiver of the Palo Alto VAMC, we have applied these capabilities in the Guardian system for intensive care monitoring. Our work on Guardian is reported in section IV of this progress report.

We have given demonstrations of the Guardian system to many colleagues in the medical AI and larger AI communities, for example to: Dr. Lawrence Fagan and his students from Stanford's Medical Computing Systems Group; Dr. Seppo Kalli, Director of Medical Signal Processing at Technical Research Centre of Finland; Dr. William Pardee and his associates from the Rockwell Science Center; Dr. Perry Thorndyke and his associates from FMC Corporation; Dr. Joseph Naser of the Electric Power Research Institute.

We have made (or plan to make) invited presentations of this research to: IEE Workshop on Expert Control Systems, Brighton England, June, 1990; International Joint Conference on Artificial Intelligence, Detroit, Aug, 1989; AI Systems in Government Conference, Washington D.C., March, 1989; AAAI Symposium on Knowledge System Development Tools, Stanford, March, 1989; Stanford SIGLunch, February, 1989; Workshop on Formal Aspects of Semantic Networks, Catalina, February, 1989; Carnegie Symposium on Architectures for Intelligence, Pittsburgh, May, 1988; Advanced Decision Systems, Palo Alto, May, 1988; Boeing Computer Services, Bellevue, WA., March, 1988; DARPA Knowledge-Based Planning Workshop, Austin, December, 1987.

The project has produced the following recent publications, including KSL 88-64, KSL 89-05, KSL 89-06, and:

Hayes-Roth, B., Hewett, M., Washington, R., Hewett, R., and Seiver, A. **Distributing intelligence within a single individual.** In L. Gasser and M.N. Huhns (Eds.) *Distributed Artificial Intelligence* Volume 2. Morgan Kaufmann, 1989.

Hewett, R., and Hayes-Roth, R. **Representing and reasoning about physical systems using generic models.** In J. Sowa (Ed.) *Formal Aspects of Semantic Networks*. Morgan Kaufmann, 1989.

Hayes-Roth, B. **Dynamic control planning in adaptive intelligent systems.** *Proceedings of the DARPA Knowledge-Based Planning Workshop*, 1989.

(2.3) Advanced Architectures

The goals and technical approach of this project, largely supported by DARPA under the Strategic Computing Program, have been discussed in previous annual reports. In the 1988 Annual Report we described the various components of the project in some detail, and reported on the current state of progress. The discussion here will be limited to reporting progress in the past year on each of the components. An overview of the entire project has recently been written (KSL 88-71), and contains a comprehensive bibliography of publications produced by the project.

SIMPLE / CARE Multiprocessor Simulation System

SIMPLE/CARE is a powerful simulation system which permits empirical studies of expert system performance on a wide class of multicomputer architectures, including quantitative measurements of system behavior. It forms the foundation for our empirical investigations of software architectures and hardware system architectures for concurrent knowledge-based systems. SIMPLE is a CAD (Computer Aided Design) system for hierarchical, multiple level specification of computer architectures and includes an associated mixed-mode, event-based simulator. CARE is a parameterized, multiprocessor array emulation specified in SIMPLE's specification languages and running on SIMPLE's simulator. Our simulation system is in use by several research groups at Stanford, and it has been ported to several external sites including NASA Ames Research Center. A videotaped tutorial was held in June, 1988, attended by representatives from industry and government, which described the CARE/SIMPLE system, as well as the LAMINA programming interface (see below). The attendees received instruction in use of the system for making measurements of the performance of various simulated multiprocessor applications.

Due to rapidly growing interest in the SIMPLE/CARE system, a major effort is now underway to port it to wider class of hardware platforms. The system is currently being reimplemented in Common Lisp and the X window system, with the Sun workstation as the initial target.

During the past year, the research effort associated with SIMPLE/CARE has largely focussed on investigations of communication protocols (KSL 88-81) and techniques for monitoring concurrent object-based applications (KSL 89-15).

LAMINA Programming Interface

LAMINA provides extensions to Lisp for studying expressed concurrency in functional programming, object oriented, and shared variable models of concurrent computation. The implementation of the support for all three computational models is based on the common notion of a stream, a data type which can be used to express pipelined operations by representing the promise of a (potentially infinite) sequence of values. LAMINA also provides system support for the management of software pipelines and dynamic structure creation, relocation, and reclamation in a multiprocessor, multi-address-space system. Algorithms and applications written in LAMINA may be run on the SIMPLE/CARE simulation system in order to study their execution on alternative multiprocessor architectures.

The development of LAMINA was completed over a year ago and is being ported to Common Lisp along with the SIMPLE/CARE system. During the past year, the shared variable model was used as the basis of a shared memory Lisp package, called QL (KSL 88-85).

CAGE and Poligon Problem Solving Frameworks

CAGE, a framework for building and executing applications as a concurrent blackboard system, was described in last year's annual report. Its development was essentially completed during the past year. The Poligon problem solving framework, for the development of Blackboard-like applications on a (simulated) multiprocessor, was also described in the last annual report. Its development is now essentially complete and documentation for the system is now available (KSL 88-69 and KSL 89-37). Reference manuals for both CAGE and Poligon are being updated, and the software itself is undergoing some minor changes to bring it in line with the documentation, in preparation for external delivery.

CAGE, Poligon and LAMINA Comparative Experiments

As described in last year's report, a series of end-to-end experiments comparing various concurrent programming systems for knowledge-based applications was conducted. The goals of these experiments are to:

- Obtain quantitative comparisons of the performance of the programming systems.
- Gain insights into how different concurrent programming models lead to different (or similar) application decomposition and organization.
- Force the refinement of the concurrent programming systems so as to better support application development.
- Gain insights into the ease or difficulty of writing application code in each of the programming systems.

The common application for these experiments is Elint [KSL 86-69], a real-time, knowledge-based system for integrating pre-processed, passively acquired radar emissions from aircraft. The Elint application was implemented in three different concurrent programming systems: LAMINA, Poligon and CAGE. During the past year we completed the experiments and documented the results (KSL 88-33, KSL 88-66, KSL 88-69, and KSL 88-80).

The AIRTRAC Application

AIRTRAC (KSL 86-20) is a knowledge-based signal interpretation and information fusion system. The system attempts to identify, track, and predict the future behavior of aircraft. In particular, it attempts to recognize aircraft which might be engaged in covert activity, for example, smuggling. The inputs to AIRTRAC are periodic radar tracking system reports, a priori, filed flight plans for some aircraft, and occasional intelligence reports about suspected covert activity. AIRTRAC is designed to be sufficiently complex and realistic to adequately test various ideas about concurrent problem solving on multiprocessor machine architectures. The AIRTRAC application involves continuous input data streams, typical of real-time signal interpretation problems. Such problems often require a level of

computational power two to three orders of magnitude beyond what is currently available. Moreover, the application uses data-driven, expectation-driven and model-driven styles of reasoning. These reasoning styles encompass a wide range of paradigms in artificial intelligence.

AIRTRAC is designed as three separate modules, called Data Association, Path Association, and Path Interpretation. The Data Association module was completed two years ago (KSL 87-34), and an initial version of the Path Association module was completed one year ago (KSL 88-41). The Path Association Module is about an order of magnitude more complex than any of our previous applications in terms of both lines of code and complexity of control, reasoning and data paths. The initial Path Association Module quantitative performance experiments yielded, at best, only about an order of magnitude speedup even on large (up to 256 processors) CARE machine architectures. This speedup was about an order of magnitude less than the best speedups obtained b earlier application experiments, such as ELINT.

The primary research question that we are investigating using the Path Association Module is whether there is an inversely proportional relationship between potential speedup using parallel execution and application complexity. If this is the case, then for most knowledge-based systems of interest the speedup via parallel execution is strongly limited. Our results to date show that for the Path Association Module the speedup limitation of about one order of magnitude is intrinsic to the application. We believe that this is primarily due to two characteristics of the application: a) Its complex control paths require significant synchronization which results in large blocks of serial execution, and b) its complex reasoning requires a fairly coarse granularity of decomposition which limits the amount of achievable parallelism.

Experiments on relaxing the control synchronization did result in improved speedup. However, in all cases the quality of solution rapidly degraded as a direct function of the amount of synchronization and was, in general, unacceptable. Experiments using finer granularity problem decompositions required increasing the amount of control synchronization, and any speedup gains due to increased execution parallelism were more than offset by the increased control serialization.

Our preliminary conclusion is that for relatively simple and well-structured applications such as ELINT, two (or possibly more) orders of magnitude speedup via parallel execution is possible. However, for complex and ill-structured applications such as AIRTRAC Path Association, speedup over a well-tuned serial program by using parallel execution is probably limited, at best, to an order of magnitude. Experiments are continuing to verify this preliminary conclusion.

During the past year we completed the design and started the implementation of a parallel, dynamic classification problem solving framework which will be used to implement the AIRTRAC Platform Interpretation Module. In this framework, a classification application is

realized as a network of sub-classification nodes. Each network node is implemented as a LAMINA object, and it executes concurrently and asynchronously. Information flow along the "links" of the network is realized by message sending. Time-tagged input data is fed into the "leading edge" nodes of the network, and the resulting information is propagated through the network to its "trailing edge" nodes. These latter nodes output results to the user whenever the classification certainty factors of any entity of interest, for example, a tracked aircraft, meaningfully change.

(2.4) Knowledge Acquisition and Machine Learning

Our research in machine learning has been ramping down, due to the departure of Professors Buchanan and Rosenbloom. However, they have continued to supervise students in our laboratory during the past year.

Inductive Rule Learning

During the past year the RL induction program was extended to learn incrementally, that is from small sets of examples presented in sequence without benefit of looking at them all together. A front-end program was written to assist in the definition of RL's starting knowledge, the so-called "half-order theory". Experiments have been started on the efficacy of combining induction and explanation-based learning. In addition, we adapted RL to be a design checker in the following sense: when a design program (or designer) proposes a new design for a device, our ability to troubleshoot it later can be partly determined by RL's ability to learn troubleshooting rules for that design.

Learning by Chunking

During the past year we completed a set of experiments evaluating a representational restriction on productions that guarantees an absence of expensive chunks, with encouraging results. We have applied our domain-independent abstraction mechanism to a set of problems in two domains (mobile robot and computer configuration), and evaluated its ability to reduce problem solving time, reduce learning time, and increase the generality of the rules learned. We have run a set of experiments which evaluate the ability of rules learned in medical diagnosis to transfer to related problems (done in a reduced-size version of NEOMYCIN-SOAR). In the area of theoretical developments and system building, we have extended our work on declarative learning to allow indexing off of arbitrary features, but in the process uncovered a new issue concerned with how to deal with multiple retrieval and discrimination.

(2.5) Symbolic Simulation

During the past year we completed the implementation of a hypothesis formation system for molecular genetics. The program, called HYPGENE, generates hypotheses to account for errors in the predicted outcomes of experiments computed by the simulation system GENSIM. HYPGENE views

hypothesis formation as a design problem. It employs design operators to reason backwards from prediction errors to alter the description of the initial conditions of the experiment to ensure that the predicted outcome of the experiment matches its observed outcome. HYPGENE generates a correct set of alternative hypotheses for two different problems encountered by biologists exploring a new mechanism of gene regulation. This research is documented in a Ph.D. dissertation, entitled "Hypothesis Formation and Qualitative Reasoning in Molecular Biology," by Peter Karp (1989).

III.A.2.4. Core System Research and Development

(1) Introduction and Overview

In this section we describe progress on our core system development and work toward a distributed AIM community. As part of our charter as a national resource, we are focusing our systems activities on producing a distributed medical research environment that is both effective for our AI-related work and can be easily reproduced at other sites. In this process, we have chosen a small number of standardized hardware and software configurations to develop and support and have tried to direct our efforts at technical areas complementary to related systems activities at other sites — we are committed to importing rather than reinventing software where possible. We continue to serve as a repository of systems information and expertise for the medical AI research community, as well as the larger computer science community. We have assisted many AIM groups in establishing local computing resources and in getting access to software available in our community (for more details, see the section on Dissemination).

One of the principal thrusts of our core systems work has been to find a technically- and cost-effective replacement for the powerful and easy-to-use computing tools of the aging DEC 2060. Our criteria for the new environment have included:

- The work environment should be modern and combine graphics, pointing, and traditional keyboard modalities of interaction, as it is expected to be the primary work environment for some years to come.
- The system should support the most powerful AI research and Lisp development environment available today, possibly involving special-purpose hardware.
- The system should support small-to-medium-size AI and Lisp-based research work without requiring special hardware.
- The cost per person should be low enough as to permit putting a machine on or near every desk and to consider the system as a potential AI delivery environment.
- The system should integrate well into a heterogeneous computing environment typical of AIM research work.
- The system should be capable of editing, organizing, and printing large documents, such as theses and books.
- The system should be capable of generating and editing state-of the art graphics.
- The heavily-used network services provided by the 2060 (e.g., wide- and local-area network access, electronic mail transmission/routing, reception, and user access, community bboards, file service, and print spooling) must be replaced.

- The design should be incrementally extendable and augmentable as new hardware and software technologies appear and as the number of users fluctuates.
- The design should be cost-effective enough as to be replicable at smaller AIM sites that wish to benefit from our experience.
- The design should permit easy data sharing and exchange with collaborators at other sites and within Stanford University.

As detailed in our report last year, we have chosen Apple Macintosh II workstations as the general computing environment for researchers and staff, TI Explorer Lisp machines (including the microExplorer Macintosh coprocessor) as the near-term high-performance Lisp research environment, and a SUN-4 as the central network server replacement for the DEC 2060. We outlined there the myriad of tasks facing us in making the transition from the central 2060 environment to the new distributed model, including selecting and integrating tools for text processing (editing, graphics, formatting, and bibliographic references), presentation graphics, printing, help facilities and distributed information access, interpersonal communication tools (EMail and BBoards), file management (storage, access, backup, and archiving), and system building tools (languages, development environments, and integration tools).

However, rather than the carefully orchestrated transition plan we had proposed and received approval from Council to implement (see last year's report for a summary), we were obliged to undertake a much more precipitous transition because of the severe funding constraints resulting from an 11% cut in our NIH/DRR grant award last year. There were two immediate consequences of this large budget cut: a) reducing our systems staff by two people (one layoff and one through attrition) and b) taking the DEC 2060 off of contract maintenance early in the grant year, thereby forcing us to close it down for routine use. These steps have had other impacts in slowing our work toward the long-term research goals we set out to achieve, both in forcing us to devote full energy to the 2060-to-SUN-4 transition approximately a year before we expected to be ready for it and in reducing the level of effort of the remaining staff for work on these difficult problems.

Because of the necessary preoccupation of most of our staff with this premature transition this past year, we were not able to convene the visiting advisory group as was recommended by BRTP to help guide our long-term research efforts. As we finally close out the 2060 chapter this summer, we will plan to assemble such a group in the early fall (September or October) to reassess our plans for the coming two years. In spite of all this unplanned redirection of our energies, we have made substantial progress this past year as summarized in the following sections.

(2) The Phase-Out of the DECSystem-20

The contract maintenance cost of the DEC 2060 was about $70,000 per year and we could not afford to continue this coverage in light of the large budget cut. Since this old-technology machine would become unreliable without regular maintenance, this forced us to transfer nearly all of our AIM community usage to the SUN-4/280 in October and November of 1988. The DECSystem-20 had been our major computer resource since February of 1983. As this machine, in turn, had replaced a DEC KI-TENEX system in use for nine years, our conversion to the UNIX based SUN-4/280 represented a major departure from a long-held approach to computing.

While some of our users already had experience with the UNIX operating system and hence had an easy time of adapting to our SUN-4 system, most others had a long history of using only the TOPS-20 operating system or its predecessor, TENEX. For these users, converting to the use of UNIX was a major obstacle. The more mnemonic command syntax of TOPS-20, together with the various command completion and ever-ready "help" features are not available in the basic UNIX system. Even the popular "man" feature for the on-line reading of UNIX manual pages provides little in the way of tutorial guidance.

A significant and urgent effort went into developing a *UNIX Users Guide for TOPS-20 Users* which has provided substantial help in navigating through the most common of commands. But, it is clear that despite its overwhelming popularity, the UNIX system fails to provide many of the user-friendly features available for many years in the TENEX and TOPS-20 environments.

The mechanics of transferring about 400 user accounts was a major but relatively straightforward task. We were fortunate in having available a program from Rand Corp. (via Rutgers University) which provides for the reading TOPS-20 Dumper tapes under UNIX. Most of the immediately-needed working files from the 2060 system were dumped to tape and loaded into the SUN-4 using this program. In addition, the Ethernet system connecting the two machines facilitated the task of transferring files and ensuring on-going access to the files remaining on the 2060. Most of this transfer was done during a four week period of intensive work.

As complicated as the transfer of the users' files was the handling of transferring the "SUMEX-AIM" name from the 2060 to the SUN-4 (including coordinating updates to all of the domain servers and host tables around the Internet), mail distribution issues, and providing continuation of BBoard facilities. Another month of planning and implementing these aspects of the transition were required, including a number of false starts because of problems in the ARPANET Network Information Center in timing the Internet name and address changes. Continuous and nearly compatible AIM community mail services were maintained through the transition by installing the Columbia University MM-C mail program on the SUN-4. This program closely duplicates the functions of the TOPS-20 COMAND JSYS

under UNIX and presents the user with a mail reader/composer interface very similar to that of TOPS-20 MM — the system that had been used by the AIM community for 10 years. At the time of our SUN-4 transition, MM-C was still in its early stages of being released and required intensive work to track down numerous bugs which were uncovered by the heavy use in our community. Nevertheless, this system, coupled with a UNIX version of the EMACS text editor, called GNUEMACS, provided a familiar setting for the most common computing functions used by AIM community members.

Once this program was in place, the 2060 mail files were passed through a preprocessor to make them UNIX-compatible. Next, we forwarded all mail directed to the old SUMEX-AIM (the 2060) to the SUN-4 (the new SUMEX-AIM). This allowed mail to be accessed during the transition. Once the transition was completed, we changed the name of the 2060 to be SUMEX-2060, and renamed the SUN-4 SUMEX-AIM. From this point onward the 2060 was no longer used for mail access. In the succeeding months, we added bulletin board functionality to MM-C so that, from the user's perspective, mail access was nearly identical to the former system.

Part of our original plan for conversion from the DECSystem-20 to the SUN-4 included moving the 2060 ARPANET interface to the SUN-4. This plan was initiated in the fall of 1988 but, before SUN Microsystems could deliver the required hardware/software interface package (and BBN could provide a replacement IMP interface) major changes in ARPANET service were being implemented. As a result, the ARPANET connection for the SUN-4 was cancelled and our Internet access is now implemented through the Bay Area Regional Research Network (BARRNet) and the NSFNet (See Wide Area Network Developments).

Another major issue in the 2060-to-SUN-4 transition has been the need to provide our users with continued access to the their large collection of archived files and to a set of permanent annual backup dumps (done January of each year) which have been collected and maintained since 1975. Early tapes are in the BSYS Archive format or TENEX Dumper format. Later ones are either TOPS-20 Archive format or TOPS-20 Dumper format.

This has required very careful planning as the directory information for these tapes resides in Archive directory files (for TENEX) and the on-line File Descriptor Blocks (FDB's) of the TOPS-20 file system. These two sets of information must be converted to simple UNIX-compatible text files to provide users with continued facilities to review and access their collections of archived files. This work has been a major undertaking and is still in progress. (See the Section on File Access, Back-up, and Archiving for more on this topic).

Our past commitment to the use of Ethernet TIPs for the connection of users to hosts proved its value in the system changes described above. Little concern had to be given to moving collections of terminal "hard-lines" from one piece of equipment to the other. Likewise, with dial-in modems attached to TIPs, users could easily select the new system when establishing